(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平１０－３２０１７２

(43)公開日　平成１０年（１９９８）１２月４日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
G06F 5/00			G06F 5/00	H	
9/318			9/30	320 A	

審査請求　未請求　請求項の数６　ＯＬ　（全１４頁）

(21)出願番号　特願平９－１２５９８６

(22)出願日　平成９年（１９９７）５月１５日

(71)出願人　０００００２３６９
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号

(72)発明者　小野　義之
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

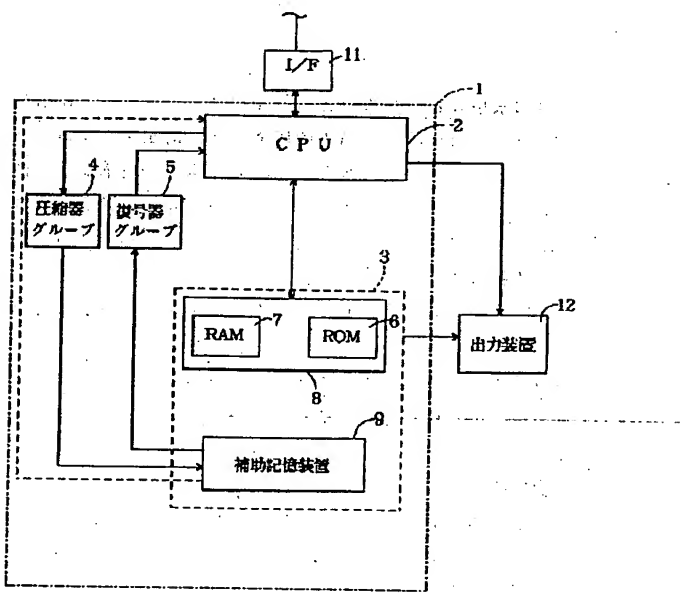
(74)代理人　弁理士　鈴木　喜三郎　（外２名）

(54)【発明の名称】プログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置

(57)【要約】

【課題】　中央処理装置を制御するためのマシンコード化されたプログラムの性質がどのようなものであろうと、一定の圧縮効果を得ること。

【解決手段】　プログラム格納装置１は、中央処理装置２と、中央処理装置２を動作させるためのマシンコードを圧縮する圧縮器グループ４と、圧縮器グループ４で圧縮されたマシンコード化されたプログラムを格納する記憶装置３と、記憶装置３内の圧縮されたプログラムを復号して中央処理装置２用のマシンコードとする復号器グループ５とを備え、命令の性質によって複数のタイプに区分されたマシンコードに対応して圧縮器グループ４のいずれか１つまたは複数を取捨選択してマシンコードを圧縮する。すなわち、中央処理装置２用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用してマシンコードを圧縮する。なお、復号時は逆のアルゴリズムを使用して復号する。



1

【特許請求の範囲】

【請求項１】　中央処理装置が制御する記憶装置に、マシンコード化されたプログラムを圧縮して格納するプログラム圧縮方法において、上記中央処理装置用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用して上記マシンコードを圧縮することを特徴とするプログラム圧縮方法。

【請求項２】　前記各命令タイプ毎に、そのマシンコードから少なくとも、レジスタフィールドおよび定数フィールドの２種類のデータを分離し、分離された各データをそれぞれ異なる符号化工程を使用して圧縮するようにしたことを特徴とする請求項１記載のプログラム圧縮方法。

【請求項３】　中央処理装置が制御する記憶装置から、圧縮されたマシンコードのプログラムを復号するプログラム復号方法において、圧縮されたマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる復号アルゴリズムを使用して、上記圧縮されたマシンコードを復号することを特徴とするプログラム復号方法。

【請求項４】　中央処理装置と、この中央処理装置を動作させるためのマシンコードを圧縮する圧縮器と、この圧縮器で圧縮されたマシンコード化されたプログラムを格納する記憶装置と、この記憶装置内の圧縮されたプログラムを復号して上記中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、上記圧縮器を複数設け、命令の性質によって複数のタイプに区分されたマシンコードに対応して上記複数の圧縮器のいずれか１つまたはいずれか複数を取捨選択してマシンコードを圧縮するようにしたことを特徴とするプログラム格納装置。

【請求項５】　少なくとも、前記マシンコードの命令フィールドを圧縮する第１の圧縮器と、前記マシンコードのレジスターフィールドを圧縮する第２の圧縮器と、前記マシンコードの定数フィールドを圧縮する第３の圧縮器とを有することを特徴とする請求項４記載のプログラム格納装置。

【請求項６】　中央処理装置と、この中央処理装置を動作させるためのマシンコードが圧縮されて格納された記憶装置と、この記憶装置内の圧縮されたプログラムを復号して上記中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、上記復号器を複数設け、命令の性質によって複数のタイプに区分されかつ圧縮されたマシンコードに対応して、上記複数の復号器のいずれか１つまたはいずれか複数を取捨選択して圧縮されたマシンコードを復号するようにしたことを特徴とするプログラム格納装置。

【発明の詳細な説明】

【０００１】

2

【発明の属する技術分野】本発明は、マイクロプロセッサ等と呼ばれる中央処理装置を動作させるためのマシンコードからなるプログラムを圧縮する方法および圧縮されたプログラムを復号する方法ならびに圧縮されたプログラムを格納する装置の改良に関する。

【０００２】

【従来の技術】中央処理装置は、その組み込まれた機器を制御するために、各種の制御や演算を行う。この制御等を行うに当たっては、付属して取り付けられる記憶装置等に格納されたプログラムに従っている。そして、このプログラムは、マシンコードとして作成されると共に、所定ビット構成に区切られて記憶装置内に格納されている。

【０００３】通常、記憶装置がパラレルに入出力できるビット数は、４ビット、８ビット、１６ビット等一定の長さに限定される。この長さは、１つのアドレスに格納できるデータのビット数に相当している。ところが、一般に、中央処理装置に対する命令は必ずしもすべて一定の長さのマシンコードで表現することができない。そこで、短いマシンコードは１つのアドレスに格納し、長いマシンコードは２つ以上のアドレスに振り分けて格納される。

【０００４】したがって、コード長の長いマシンコードは、短いものに比べて読み出しに時間がかかり、このようなマシンコードが多いとプログラムの実行速度が低下してしまう。また、コード長の長い命令が多く含まれるプログラムは、これを格納するために大容量の記憶装置を必要とする難点もある。

【０００５】これを解決するために、プログラムをマシンコード化して記憶装置に格納する場合において、プログラムに含まれる命令のうち、マシンコード化したときそのコード長が比較的長いものであって、そのプログラム中に比較的多く含まれている特定の命令について、そのマシンコードをこれより短い圧縮マシンコードに置換した後、記憶装置に格納する技術が知られている（特開昭６１－２０１３３５号公報参照）。

【０００６】また、所定回数連続するマシンコードを検出し、これを特定形式の圧縮コードに置き換えて記憶装置に格納するようにし、連続したマシンコードの格納領域を大幅に低減した装置も知られている（特開平３－１１０６４５号公報参照）。

【０００７】他方、“０”と“１”からなる２値信号を圧縮する情報理論の世界では、算術符号化方式と呼ばれるものが知られている。この算術符号化方式は、エントロピー符号化方式であり、本質的に可逆符号化（ロスレス）の性質を持つものである。そして、その原理は、エライアスの符号化として知られている無記憶情報源に対する理想的符号化方式を実用可能な形に再編成したものとなっている。すなわち、算術符号とは、“０”と“１”の直線上の対応区間を各シンボルの生起確率に応

3

じて不等長に分割していき、対象シンボル系列を対応する部分区間に割り当て、再帰的に分割を繰り返していくことにより得られた区間内に含まれる点の座標を、少なくとも他の区間と区別できる２進小数で表現してそのまま符号とするものである。

【０００８】この算術符号化方式は、有限個の情報源シンボルに特定の符号語を対応させるブロック符号に比べ、符号器の規模が必要メモリ量などのハードウェアが小さくて済むこと、高い効率を期待できることおよび適応符号化が容易なこと等の利点がある。このこと等から、２値信号を扱う情報理論の世界では、この算術符号化方式がその情報の持つエントロピーに最も近いレベルに圧縮できるとされ、最も効率の良い符号化方式と言われている。なお、この算術符号化方式は、特に、後述するマルコフ情報源の符号化に適するものとなっている。

【０００９】この算術符号化方式として、Ｑコーダ、算術符号型ＭＥＬコード、Ｍｉｎｉ－Ｍａｘコーダ等が提案されている。そして、これらの算術符号を改善したものとして、ＱＭコーダと呼ばれている方式が知られている。このＱＭコーダは、カラー静止画符号化標準（ＪＰＥＧ）および２値画像符号化標準（ＪＢＩＧ）の両標準において、共通に使用されている。なお、このＱＭコーダは、２値情報源用の符号であり、ＪＰＥＧのような多値情報源の符号化にあたっては、その多値情報源を２値化するための前処理を必要としている。このような場合、符号化すべき２値シンボル数は増大するが、多値情報源としての情報量を増大させることなしに２値系列に変換することが可能となっている。

【００１０】このＱＭコーダは、ＪＰＥＧおよびＪＢＩＧの規定の中にその仕組みについて詳細に述べられているが、ここでは後述する本発明を理解する参考として、その概要を図９に基づき簡単に説明する。なお、算術復号型のエントロピー復号器の構成は、エントロピー符号器の構成と実質的に同一であるので、ここではその説明は省略する。

【００１１】この算術符号型のエントロピー符号器となるＱＭコーダ１０１は、算術演算部１０２と、状態記憶器として機能する発生確率生成手段１０３とを含んで構成される。この発生確率生成手段１０３内には、符号化に必要なシンボル発数確率を決定するために必要な状態パラメータテーブルが書き込まれている。上記の状態パラメータは、入力される状態信号１０６によって特定される。そして、この状態信号１０６によって特定された状態パラメータのテーブルに対し、発生確率手段１０３の発生確率演算パラメータが算術演算部１０２へ向け出力される。

【００１２】算術演算部１０２は、このようにして入力される発生確率に基づき、エントロピー符号化を行い、入力されるデータ１０４を符号化データ１０５に圧縮し、符号化して出力する。そして、符号化されるデータ

4

１０４の値により、状態信号に対する発生確率を再計算し、演算パラメータ更新値が、発生確率生成手段１０３へ入力する。この更新結果が次データの発生確率としてテーブルに記憶されることで、ＱＭコーダ１０１の圧縮効率が向上することとなる。なお、入力されるデータは２値化された信号であるが、その信号としては、文字情報や画像データや順位コード変換により変換された色順位データ（特開平６－２７６０４１号参照）等が代表例である。図１０に、このような技術を多値情報源の代表例である画像データに用いた従来の符号化システム１５０および復号化システム１６０を示す。この符号化システム１５０は、ラインバッファ１５１と、エントロピー符号器１５２とを含むものである。入力されるインデックスの画素データ１００Ａは、ラインバッファ１５１およびエントロピー符号器１５２へ入力される。この画素データ１００Ａは、図１１に示すように、いずれもラスタースキャンされ水平走査順に順次画素データとして入力される。

【００１３】符号化システム１５０中のラインバッファ１５１は、参照画素生成手段として、既に入力された画素データ１００Ａから、符号化対象画素Ｘに対する参照画素データＡ，Ｂ，Ｃ，Ｄを作成する。すなわち、ラインバッファ１５１は、画像をスキャンするときにｎライン（１～５ライン程度が多い）分の履歴を記憶しておく。そして、符号化対象画素Ｘの画素データ１００Ａが入力されるごとに、この直前の画素Ａと、周辺の画素Ｂ，Ｃ，Ｄとからなる一連の画素データを参照画素データ１１０としてエントロピー符号器１５２へ向けて出力する。

【００１４】このエントロピー符号器１５２は、ＱＭコーダ１０１が使用されているが、ハフマン符号化などの手法を用いて形成したものとしても良い。そして、参照画素データ１１０（一種のコンテキスト）を状態信号として用い、対象画素データ１００Ａを符号化データ２００に変換出力する。

【００１５】一方、復号化システム１６０は、ラインバッファ１６１とエントロピー復号器１６２を含んで構成される。ここにおいて、ラインバッファ１６１とエントロピー復号器１６２は、入力される符号化データ２００を符号化システム１５０のラインバッファ１５１、エントロピー符号器１５２とは全く逆の手順で復号化出力するように形成されている。

【００１６】このようにして、符号化システム１５０と、復号化システム１６０とは、互いに全く逆のアルゴリズムを用いて、画素データ１００Ａを符号化データ２００に符号化し、さらにこの符号化データ２００を画素データ１００Ｂに復号化して出力することができる。したがって、このシステムは、各種用途に幅広く用いられている。

【００１７】しかし、このようなエントロピー符号器１

5

５２およびエントロピー復号器１６２では、参照画素データの状態数に対応した数の符号化パラメータテーブルが必要となる。このため、圧縮率を高めるために参照画素数を大きくとればとるほど、符号化および復号化のパラメータテーブルが大きくなる。このため、エントロピー符号器１５２およびエントロピー復号器１６２が大型化かつ高価となってしまうという問題がある。

【００１８】このような問題に対し、エントロピー符号器１５２およびエントロピー復号器１６２の中に縮退した状態数に応じてパラメータテーブルを小さくさせる技術が知られている。

【００１９】この状態数を縮退するシステムの特徴は、図１２に示すように、図１０の符号化システム１５０や復号化システム１６０と同様にエントロピー符号器１５２およびエントロピー復号器１６２に参照画素データ１１０を状態信号として入力するわけであるが、その入力に際し、その状態信号１４０を、ラインバッファ１５１，１６１から出力される参照画素データ１１０を縮退する状態縮退器１５３，１６３によって生成する点にある。

【００２０】この状態縮退器１５３、１６３は、入力される参照画素データ１１０を、より少ないビット数の状態信号１４０に縮退し、対応するエントロピー符号器１５２およびエントロピー復号器１６２へ向け出力するように構成されている。なお、予測器１５４，１６４は、処理する画素がカラーである場合、それぞれカラーシンボルの出現頻度に基づいてカラー画素データを色順位に変換するためおよびその逆を行うための色順位テーブルをそのメモリーに保有しているものである。

【００２１】なお、縮退とは、縮退後の状態数に、元の状態を分類する操作である。この分類は、分類後のエントロピー（１つのシンボルを表示するための平均情報量）が最少となるように、その組み合わせを選択して行う。そして、縮退後の状態数、すなわち、分類された後の状態数に対して識別ビットを付加する。これが状態信号１４０である。

【００２２】なお、入力されてくるデータ、色順位データ１２０を生成するために、予測器１５４，１６４内に色順位テーブルが配置されている。この色順位テーブルの一例として、図１３に示されるものが知られている（特開平６－２７６０４１号公報参照）。この例では、符号化対象画素Ｘに対しての色順位テーブルを決める際、２次元的な周辺画素データＲ０，Ｒ１，Ｒ２，Ｒ３を上位の色順位データとして使用し、符号化対象画素Ｘと同一ラインの１次元テーブルを下位の色順位データとして使用するものである。このとき、１次元テーブルから周辺画素データＲ０，Ｒ１，Ｒ２，Ｒ３のカラーシンボルを除去した後、上位と下位の色順位データをドッキングさせ符号化対象画素Ｘの色順位テーブルとしている。

6

【００２３】具体的にどのように色順位テーブルができあがるかを図１３に基づき説明する。符号化されるカラーシンボルが１６色の場合を考える。仮に、色順位を図１３（Ａ）に示すように、各画素の位置Ｒ０，Ｒ１，Ｒ２…Ｒ８… で固定したとき、それぞれのカラーシンボルが図１３（Ｂ）に示すように、Ｃ４，Ｃ３，Ｃ６，Ｃ５，Ｃ２，Ｃ２…のとき、できあがる最新出現表となる色順位テーブルは、図１３（Ｃ）に示すようになる。すなわち、最上位はＲ０のＣ４，２番目はＲ１のＣ３，３番目はＲ２のＣ６，４番目はＲ３のＣ５，５番目はＲ４のＣ２，６番目はＲ５にあるＣ２となるが、Ｃ２は既に発生しており、さらにＲ６のＣ４も既に発生しているので、第６番目はＲ７のＣ０となる。このようにして既に上位にある色すなわち、Ｒ０～Ｒ３に出現するカラーシンボルを除いた色順位データがＲ０～Ｒ３のデータに加わり、１６色のカラーシンボルの第１番目から１６番目までが決められる。

【００２４】

【発明が解決しようとする課題】このように、中央処理装置を制御するマシンコードを圧縮するには、特開昭６１－２０１３３５号公報や特開平３－１１０６４５号公報に開示されるように、プログラムの特徴を検出して、圧縮する方法がある。また、その圧縮に利用する技術としては、上述したように、ＱＭコーダ１０１等のエントロピー符号器、複数のレジスタからなるＦＩＦＯを利用した、いわゆる先頭移動処理（Move to Front）の最新出現表利用の技術、色順位変換等の順位コード変換技術、マルコフモデル等の状態縮退技術等が採用される。

【００２５】しかしながら、本発明者の実験によれば、ＱＭコーダ１０１や最新出現表や順位コード変換等の各方式のみでは、中央処理装置のマシンコードは十分圧縮されないことが判明した。これは、その性質を無視して、単に８ビット等の単位で圧縮しているためである。この１バイト単位の圧縮は、ＬＨＡ等のＬＺ系の圧縮の際にも行われている。

【００２６】また、特開昭６１－２０１３３５で開示される技術は、マシンコードのコード長の長い命令が多く含まれる特殊なプログラムには十分な効果があるものの、通常のプログラムに対しては圧縮効果はほとんど生じない。また、特開平３－１１０６４５の技術は、特定回数連続するコードを多く含むものに対しては、かなり効果があるものの、連続するコードがそれ程ないものについては、やはり圧縮の効果は出てこない。

【００２７】本発明は、中央処理装置を制御するためのマシンコード化されたプログラムの性質がどのようなものであろうと、一定の圧縮効果を得ることができる新規なプログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置を提供することを目的とする。

【００２８】

【課題を解決するための手段】かかる目的を達成するた

め、請求項１記載のプログラム圧縮方法では、中央処理装置が制御する記憶装置に、マシンコード化されたプログラムを圧縮して格納するプログラム圧縮方法において、中央処理装置用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用してマシンコードを圧縮している。

【００２９】このように、中央処理装置を動作させる各コードが意味する命令の性質によって、複数のタイプにマシンコードを区分して、異なるアルゴリズムで圧縮するようにしているので、命令の性質に合った最適な圧縮が可能となる。このため、圧縮率が向上し、必要な記憶容量を低減させることができる。

【００３０】また、請求項２記載の発明では、請求項１記載のプログラム圧縮方法において、各命令タイプ毎に、そのマシンコードから少なくとも、レジスタフィールドおよび定数フィールドの２種類のデータを分離し、分離された各データをそれぞれ異なる符号化工程を使用して圧縮している。

【００３１】このため、マシンコードのタイプ毎の区分に加え、各フィールド毎に専用の圧縮器で符号化するので、一層、圧縮率が高いものとなる。そして、特に、ビット数が多くなるレジスタフィールドと定数フィールドをそれぞれ専用の符号化工程によって処理できるので他のフィールドの専用化に比べ圧縮率の面で有利となる。

【００３２】また、請求項３記載のプログラム復号方法では、中央処理装置が制御する記憶装置から、圧縮されたマシンコードのプログラムを復号するプログラム復号方法において、圧縮されたマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる復号アルゴリズムを使用して、圧縮されたマシンコードを復号している。

【００３３】このように復号に際し、その命令の性質に合った復号アルゴリズムを使用できるので、復号効率が向上する。また、圧縮されたマシンコードが保存される記憶装置の容量を低減させることができる。

【００３４】さらに、請求項４記載の発明では、中央処理装置と、この中央処理装置を動作させるためのマシンコードを圧縮する圧縮器と、この圧縮器で圧縮されたマシンコード化されたプログラムを格納する記憶装置と、この記憶装置内の圧縮されたプログラムを復号して中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、圧縮器を複数設け、命令の性質によって複数のタイプに区分されたマシンコードに対応して複数の圧縮器のいずれか１つまたはいずれかの複数を取捨選択して、マシンコードを圧縮するようにしている。

【００３５】このように、中央処理装置を動作させる各マシンコードが意味する命令の性質によって複数のタイプにマシンコードを区分して、各タイプ毎に対応する圧縮器を使用しているので、命令の性質に合った最適な圧縮が可能となる。このため、この装置を使用すると、プログラムの性質に拘わらず、圧縮率が向上する。また、記憶容量も減少させることができ、この装置の高機能化も達成し易くなる。

【００３６】加えて、請求項５記載の発明では、請求項４記載のプログラム格納装置において、少なくとも、マシンコードの命令フィールドを圧縮する第１の圧縮器と、マシンコードのレジスターフィールドを圧縮する第２の圧縮器と、マシンコードの定数フィールドを圧縮する第３の圧縮器とを設けている。

【００３７】このため、マシンコードのタイプ毎の区分に加え、各マシンコードのフィールド毎に専用の圧縮器で符号化するので、一層、圧縮率が高いものとなる。

【００３８】また、請求項６記載の発明では、中央処理装置と、この中央処理装置を動作させるためのマシンコードが圧縮されて格納された記憶装置と、この記憶装置内の圧縮されたプログラムを復号して中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、復号器を複数設け、命令の性質によって複数のタイプに区分されかつ圧縮されたマシンコードに対応して、複数の復号器のいずれか１つまたはいずれか複数を取捨選択して圧縮されたマシンコードを復号している。

【００３９】このように、中央処理装置を動作させる圧縮された各マシンコードが意味する命令の性質によって複数のタイプにその圧縮されたマシンコードを区分して各タイプ毎に対応する復号器を使用しているので、命令の性質に合った最適な復号が可能となる。このため、この装置を使用すると、プログラムの性質に拘わらず復号効率が向上する。

【００４０】本発明のプログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置では、中央処理装置が使用するマシンコードのくせを利用して、そのマシンコードを圧縮したり、復号したりしている。すなわち、中央処理装置の命令タイプによって、命令コード（一般にオペコードと呼ばれる）の後の部分の使われ方が違ってくるので、その命令タイプを判断して、命令タイプ毎に最適な圧縮方法や復号方法を採用している。よって、処理されるプログラムがどのような性質をもっていたとしても、一定の圧縮効果が発生し、復号効率も向上する。しかも、その圧縮率は、実験結果によると、ＬＺ系の圧縮方式と同等レベルないしはそれ以上の圧縮率を得ることが出来るものとなっている。

【００４１】

【発明の実施の形態】以下、本発明の実施の形態のプログラム格納装置１ならびにプログラム圧縮方法およびプログラム復号方法の第１の実施の形態について、図１から図６に基づき説明する。

【００４２】このプログラム格納装置１は、中央処理装

9

置（以下、ＣＰＵという）２と、記憶装置３と、ＣＰＵ２を制御するプログラムを圧縮して記憶装置３に記憶させる複数の圧縮器からなる圧縮器グループ４と、記憶装置３に書き込まれた圧縮データを読み出して復号する複数の復号器からなる復号器グループ５とから主に構成される。ここで、記憶装置３は、リード・オンリ・メモリ（以下ＲＯＭという）６およびランダム・アクセス・メモリ（以下ＲＡＭという）７からなる主記憶装置８と、大量の情報を蓄える補助記憶装置９とからなる。

【００４３】なお、このプログラム格納装置１には、ＣＰＵ２が、図示しないキーボードや外部記憶装置等と信号をやり取りするためのインターフェイス（Ｉ／Ｆ）１１と、ＣＲＴや液晶パネル等で構成されＣＰＵ２の制御演算結果を表示する出力装置１２等が接続されている。また、圧縮器グループ４は、ハードではなく、いわゆるエンコードプログラムとして形成しても良く、復号器グループ５も同様に、デコードプログラムとして形成しても良い。

【００４４】ここで使われているＣＰＵ２の命令は、１６ビット・フォーマットと３２ビット・フォーマットの２種類となっている。そして、１６ビット命令には、２項演算、制御、条件分岐などがあり、３２ビット命令にはロード／ストア、Ｉ／Ｏ操作、１６ビット・イミーディエトを扱う命令、ジャンプ・アンド・リンクなどがある。なお、一部の命令では、未使用フィールドが発生するが、それらは将来の拡張用で「０」に固定される。実際に命令がメモリに格納されるときは、次のように配置される。すなわち、各命令形式の下位部分（ビット０を含む）は、下位アドレス側に、各命令形式の上位部分（ビット１５またはビット３１を含む）は、上位アドレス側に配置される。

【００４５】そして、各命令は、その性質によって図２に示すように、９種類のフォーマットでかつ計１６種類のタイプに区分される。第１番目のＡグループに属するものは、図２（Ａ）に示すように、いわゆるreg-reg命令形式となっている。このタイプのものは、６ビットのオペコード・フィールド（＝命令フィールド）とオペランド指定に２つの汎用レジスタ指定フィールド（＝レジスタフィールド）をもつ命令形式となっており、１６ビット長命令となっている。第２番目のＢグループに属するものは、図２（Ｂ）に示すように、いわゆるimm-reg命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、５ビットのイミーディエト・フィールド（＝定数フィールド）と、１つの汎用レジスタ指定フィールドをもつ命令形式となっており、１６ビット長命令となっている。なお、この実施の形態では、定数フィールドとは、命令フィールド、レジスタフィールドおよび後述する条件フィールドの計３つのフィールド以外のフィールドを指すものとしている。

【００４６】第３番目のＣグループに属するものは、図２（Ｃ）に示すように、いわゆる条件分岐命令形式となっている。このタイプのものは、３ビットのオペコード・フィールド（＝命令フィールド）と、４ビットの条件コードフィールド（＝条件フィールド）と、９ビットの分岐ディスプレースメント・フィールド（＝定数フィールド）と、１ビットのサブオペコートフィールド（＝定数フィールド）をもつ命令形式となっており、１６ビット長命令となっている。なお、９ビットの分岐ディスプレースメント・フィールドでは、ビット”０”は「０」とみなし、指定しないようになっている。また、第４番目のＤグループに属するものは、図２（Ｄ）に示すように、いわゆる中距離ジャンプ命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、２６ビットのディスプレーメント・フィールドをもつ中距離分岐命令形式となっており、３２ビット長命令となっている。

【００４７】第５番目のＥグループに属するものは、図２（Ｅ）に示すように、いわゆる３オペランド命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、２つの汎用レジスタ指定フィールドと、１６ビット・イミーディエト・フィールドをもつ命令形式となっており、３２ビット長命令となっている。さらに、第６番目のＦグループに属するものは、図２（Ｆ）に示すように、いわゆるロード／ストア命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、２つの汎用レジスタ指定フィールドと、１６ビット・ディスプレースメント・フィールドをもつ命令形式となっており、３２ビット長命令となっている。

【００４８】さらに、第７番目のＧグループに属するものは、図２（Ｇ）に示すように、いわゆる拡張命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、２つの汎用レジスタ指定フィールドと、６ビットのサブオペコード・フィールド（＝定数フィールド）をもつ命令形式となっており、３２ビット長命令となっている。また、第８番目のＨグループに属するものは、図２（Ｈ）に示すように、いわゆる３レジスタ・オペランド命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、３つの汎用レジスタ指定フィールドと、６ビットのサブオペコード・フィールドをもつ命令形式となっており、３２ビット長命令となっている。なお、図２（Ｇ）中で示される「ＲＦＵ」は、「Reserved for Future Use」の略で、本実施の形態では、未使用部分とみなして圧縮および記憶せずに捨ててしまっている。

【００４９】第９番目のＩグループに属するものは、図２（Ｉ）に示すように、いわゆるオペランドなし命令形式となっている。このタイプのものは、６ビットのオペコード・フィールドと、１ビットのサブオペコード・フィールドをもつ命令形式となっており、１６ビット長命

10

令となっている。その後のＪグループからＰグループのものは、それぞれ前述の９種類のフォーマットのいずれかと同様であるが、その命令の性質によって違うグループに区分されるものである。

【００５０】以上のように、ＣＰＵ２の命令がその性質によって複数のタイプに区分される。そして、その各タイプによって使用される圧縮器が異なるものとなる。すなわち、圧縮器グループ４は、第１の圧縮器となる命令フィールド（オペコード・フィールド）用の圧縮器と、第２の圧縮器となるレジスタフィールド（汎用レジスタ指定フィールド）用の圧縮器と、第３の圧縮器となる条件フィールド（条件コード・フィールド）用の圧縮器と、第４の圧縮器となる定数フィールド用の圧縮器の計４つの圧縮器から構成され、命令タイプによって使用する圧縮器を図３に示すように異ならせている。

【００５１】なお、各圧縮器としては、ここでは、算術符号式のエントロピー符号器、具体的には図９で示したようなＱＭコーダを使用しているが、他の圧縮器を使用しても良い。同様に各復号器は算術復号型のエントロピー復号器を使用しているが、他の復号器を使用しても良い。なお、第１の圧縮器および第１の復号器は、命令フィールドが６ビットとなっているが、命令タイプとしては１６種類に区分されているため、４ビットで全タイプを現すことができ、４ビット幅を有するＱＭコーダとしている。また、第２の圧縮器および第２の復号器は、レジスタフィールドが５ビットとなっているので、５ビット幅を有するＱＭコーダとしている。同様に条件フィールド用の第３の圧縮器および第３の復号器は、４ビット幅のＱＭコーダで、定数フィールド用の第４の圧縮器および第４の復号器は、１ビット幅のＱＭコーダとしている。

【００５２】次に、このように構成されるプログラム格納装置１の圧縮格納動作について、図４等に基づき説明する。

【００５３】まずは、ＣＰＵ２がインターフェイス１１を介してまたは補助記憶装置９内に取り込まれていた格納すべきプログラムを読み出す。インターフェイス１１を介する場合の例としては、通信等により特定のプログラムをダウンロードする場合やキーボードによって直接プログラムを生成する場合等がある。そして、そのプログラムを記憶装置３の一部を構成する補助記憶装置９に保存するとき、この圧縮処理がスタート（ステップＳ１）する。

【００５４】最初、プログラムに書かれたＣＰＵ２の命令タイプがＡグループに属するものか否か判断する（ステップＳ２）。そして、Ａグループに属するものであるときは、Ａグループの圧縮器、具体的には図３に示すように、命令フィールド用の第１の圧縮器と、レジスタフィールド用の第２の圧縮器とを使用して圧縮する（ステップＳ３）する。

【００５５】この圧縮の際、過去のデータの情報を各ＱＭコーダにコンテキスト（＝状態信号）として入力する。これによって圧縮率をさらに高めている。なお、このコンテキスト入力は、各圧縮の際、すべてのＱＭコーダに入力するようにしている。ただし、後述するように、コンテキストを入力しない場合でも相当な圧縮率を得ることができる。

【００５６】そして、圧縮されたそのマシンコードを記憶装置３、この実施の形態では、補助記憶装置９に書き込む（ステップＳ４）。その後、そのプログラムが終了か否か判断して（ステップＳ５）、終了の場合はストップする（ステップＳ６）。終了でないときは、次のマシンコードを処理するためステップＳ２に戻る。

【００５７】一方、最初の命令タイプがＡグループに属さないときは、Ｂグループに属するか否か判断して（ステップＳ７）、Ｂグループに属するときは、Ｂグループの圧縮器によってそのマシンコードを圧縮する（ステップＳ８）。具体的には、図３に示すように、命令フィールド用の第１の圧縮器と、レジスタフィールド用の第２の圧縮器と、定数フィールド用の第４の圧縮器とを使用して圧縮する。そして、圧縮されたマシンコードを同様に記憶装置３に書き込む（ステップＳ９）。この後、プログラムが終了か否か判断して（ステップＳ５）、終了であればストップし（ステップＳ６）、終了しない場合は、次のマシンコードを処理するためステップＳ２に戻る。

【００５８】最初の命令タイプがＡグループにもＢグループにも属さない場合、Ｃグループに属するか否か判断して（ステップＳ１０）、Ｃグループに属するときは、Ｃグループの圧縮器によってそのマシンコードを圧縮する（ステップＳ１１）。具体的には、図３に示すように、命令フィールド用の第１の圧縮器と、条件フィールド用の第３の圧縮器と、定数フィールド用の第４の圧縮器とを使用して圧縮する。そして、圧縮されたマシンコードを同様に記憶装置３内の補助記憶装置９に書き込む（ステップＳ１２）。以下、上述したステップと同様なステップを繰り返して圧縮し格納していく。

【００５９】なお、この圧縮においては、特に、レジスタフィールドに書かれるレジスタ番号や定数フィールドに書かれる分岐命令の圧縮の際に効果が大となる。これは、レジスタ番号や分岐命令部分がビットを多く必要とする部分であるためである。特に、レジスタ番号は、同じものがたびたび現れることが多いため、一度現れてきたものを複数個、具体的にはこの実施の形態では２個残しておき、コンテキストとして入力させることにより圧縮率が高まることとなる。なお、２個ではなく、３個以上としても良く、また、このコンテキストとしての入力の際、図１３に示すような最新出現表を作成し、利用すると一層、圧縮率が向上する。

【００６０】また、分岐命令の場合、現在位置から現し

た相対位置、例えば「現在位置から２０番地前の位置へ移れ」等の相対位置で表現されることが多い。このような場合、プログラムが進んで、現在位置がどんどん進んでいくと、戻る番地は同じであっても表現は異なるものとなる。例えば、先の例で言えば、プログラムが３番地進んでいると、「現在位置から２３番地前の位置へ移れ」という表現となる。これは数字が変わることになり圧縮にとって不利な表現である。よって、この実施の形態では行っていないが、プログラムが進むごとに現在位置を覚えておいて、すなわちプログラムが進むごとに現在地を補正して、元のアドレスを常に圧縮するようにするのが好ましい。

【００６１】なお、この分岐命令の場合においても、過去の履歴を覚えておいて、その情報をコンテキストとして入力するようにしても良い。この際、相対番地として覚えておくか、絶対番地として覚えておくかは、適宜選択することができる。また、過去の履歴としては、先程のように先頭移動処理を行うようにしても良い。

【００６２】また、オペコード（ｏｐｃｏｄｅ）の部分は６ビットで表現されるが、６ビットすなわち$2^6=64$種類の命令があるがというとそうではなく、６４種類未満の命令となっている。通常の圧縮であると、このあり得ない命令も通常の処理によって圧縮するが、この実施の形態では命令タイプの性質によって区分しているので、あり得ない命令が出てくると、ストップさせるか通常の圧縮を行うようにできる。

【００６３】さらに、未定義命令が発生するような場合、その命令を認識すると、その後のビットは切り捨ててしまうようなこともできる。さらには命令タイプによっては、その後に生ずるビットが確定しているものを生ずることがある。このような場合、例えば、テーブルを持っておいて、対応するその旨の符号のみを記憶させておき、復号するときに、そのテーブルから該当する確定ビットを持ってくるようにすることもできる。

【００６４】この実施の形態のプログラム格納装置１における圧縮率は、図５に示すとおりとなっている。すなわち、コンテキストを利用せず、単に分類したときでも約７５．０％の圧縮率を示している。その値は、ＬＺ系の基本アルゴリズムを使用したときの値よりは６％程度悪くなっているが、それでもかなりの圧縮率となっている。

【００６５】一方、コンテキストを図６に示すような形で利用したときは６９．６％となり、ＬＺ系とほとんど同じとなっている。このコンテキスト利用は、単純に直前の過去のデータの情報を利用するだけのもので、しかも図６に示すように、レジスタフィールドと定数フィールドのみしか利用していない。このような利用程度でもかなりの圧縮率となっている。よって、ＱＭコーダにマシンコードを入れる前に、先に示した先頭移動処理や相対位置の記憶等、各種の改善策によって処理すれば、さらに一層良い圧縮率が得られることが想定される。

【００６６】なお、命令フィールドは、今回、その性質に併せて１６種類に人手によって分けたが、統計をとってかつ自動的に分けるようにすることが好ましい。また、レジスタフィールドは、先頭移動処理を行ってからＱＭコーダに入れるようにするのが好ましい。さらに、条件フィールドは、オペコードの条件とからめて予測するのが好ましい。定数フィールドのアドレスに対しては、相対番地を絶対番地に変えて、それを記憶させ一種の先頭移動処理をやらせるようにしたり、ハフマンコードのような処理の仕方を行うようにしても良い。

【００６７】また、プログラム中に生のデータがある場合は、そのプログラム中にフラグを立て、そのフラグを検知して汎用のアルゴリズムに切り替えるようにするのが好ましい。この場合、そのデータ部分については、先頭移動処理や単なるＱＭコーダやＬＺ系の圧縮方法を採用するようにする。また、コンパイラによってデータ部分を分離するようにしても良い。

【００６８】なお、ネットワーク中でプログラムを移動させたり、アップロードや、ダウンロードするときに、課金がなされるネットワーク部分は圧縮して、小さい容量にして移動させ、その他の部分は復号して移動させるようにしても良い。例えば、フロキシサーバにこの圧縮復号機能を持たせ、自動的にフィルタリングするようにする技術を採用できる。このようにすることにより、大容量のプログラムを短時間で送信したり、アップロードやダウンロードを短時間かつ低費用で行えるようになる。しかも、作業者は、圧縮する際、特別な作業を必要としないので、煩わしさがなくなる。

【００６９】なお、このＣＰＵ２の命令タイプは、図２のように、１６種類に区分できるが、マシンコードのフォーマットや性質は、各ＣＰＵやコンパイラによって全く異なるため、ＣＰＵが異なればその区分けの仕方も全く異なるものとなる。すなわち、各ＣＰＵ毎に異なる処理工程および異なる圧縮器や復号器が必要となる。

【００７０】次に、本発明の第２の実施の形態のプログラム格納装置２１を図７に基づいて説明する。このプログラム格納装置２１で使用している圧縮方法や好ましい圧縮方法については、第１の実施の形態のプログラム格納装置１と同様である。また、説明に当たり、第１の実施の形態のプログラム格納装置と同一部材には同一符号を付して、その説明を省略または簡略化することとする。

【００７１】この第２の実施の形態のプログラム格納装置２１では、圧縮器グループ４と復号器グループ５をＣＰＵ２と主記憶装置８との間に配置している。このため、ＲＯＭ６に書き込まれる、ＣＰＵ２と密接不可分なプログラムが圧縮して保存されたり、インターフェイス１１等から入力され一旦ＲＡＭ７に書かれるマシンコードも圧縮して保存されることとなる。このため、ＲＯＭ

15

６やＲＡＭ７の容量を小さなものにでき、同じ容量のＲＯＭ６やＲＡＭ７を採用すると、大容量のプログラムの処理が可能となる。

【００７２】この第２の実施の形態でも、圧縮器グループ４や復号器グループ５をそれぞれハードではなく、エンコードプログラムやデコードプログラムに置き換えても良い。この場合、各プログラムを、圧縮されたプログラムが格納されるＲＯＭ６とは別のＲＯＭに書き込んだり、ＲＯＭ６を分割して、分割された部分に書き込んでおくのが好ましい。そして、その別のＲＯＭ内のエンコードプログラムやデコードプログラムを使用して圧縮したり復号したりするようにする。また、ＲＯＭ６内に書き込まれる当初のプログラムは圧縮せず通常の形態とし、ＲＡＭ７に保存されるマシンコードのみを圧縮するようにしても良い。

【００７３】次に、本発明の第３の実施の形態のプログラム格納装置４１を図８に基づいて説明する。このプログラム格納装置４１で使用している圧縮方法や好ましい圧縮方法についても第１の実施の形態のプログラム格納装置１と同様である。また、説明に当たっては、第１の実施の形態のプログラム格納装置と同一部材には同一符号を付して、その説明を省略または簡略化することとする。

【００７４】この第３の実施の形態のプログラム格納装置４１では、ＣＰＵ２と主記憶装置８との間に復号器グループ５のみを配置している。ＲＯＭ６へのプログラムの圧縮書き込みは、ＣＰＵ２と同様な動作をするプログラムおよびエンコードプログラムによって図示しない別ルートで行う。このため、ＲＯＭ６には、この発明の方法で圧縮されたプログラムが既に格納されている。ＣＰＵ２は、復号器グループ５によって復号されたプログラムによって動作するようになる。このようなプログラム格納装置４１は、特定の用途に特化されたＣＰＵ２、例えば電気炊飯器や電気冷蔵庫等の家庭用電気製品や、ロボット等の工業製品等に使用されるＣＰＵ２に適用して好適となる。

【００７５】なお、上述の各実施の形態は、本発明の好適な実施の形態の例であるが、これに限定されるものではなく、本発明の要旨を逸脱しない範囲において、種々変形実施可能である。例えば、第１の実施の形態のプログラム格納装置１と、第２の実施の形態のプログラム格納装置２１とを合わせた構成、すなわち圧縮器グループ４と復号器グループ５を、ＣＰＵ２と補助記憶装置９との間およびＣＰＵ２と主記憶装置８との間の両部分に配置するようにしても良い。

【００７６】また、本発明は、命令の分類分けが１６種類のようにある程度少ない数に分けることができるものに適用して好適であるが、３０種類や６０種類のように多くの種類に分類分けされるものにも適用できる。

【００７７】また、上述の実施の形態では、４つのＱＭコーダを使用したが、４つ以外の数としたり、少なくとも、ビット数が多くなりがちなレジスタフィールド用および定数フィールド用の２つの圧縮器や復号器を使用することにより、プログラムの圧縮率を相当高めることができる。加えて、ＱＭコーダ以外の各種の圧縮や復号のアルゴリズムを採用することができる。

16

【００７８】さらには、圧縮器や復号器を１つとし、命令タイプの性質によって区分した表をコンテキストとして利用するだけでも圧縮率は向上する。また、条件フィールドを圧縮する際、命令フィールドの性質とからませて一種の状態縮退器（図１２参照）を形成してその状態をコンテキストとしてＱＭコーダ等の圧縮器に入力するようにしても良い。さらには、分離されたフィールドの性質によっては、圧縮せず、そのまま記憶させるようにしても良い。また、各圧縮器や各復号器に、特定データについては、圧縮や復号をせず、入力されてきたデータをそのまま素通しさせる、いわゆる素通し機能を持たせるようにしても良い。

【００７９】

【発明の効果】以上説明したように、請求項１および２記載のプログラム圧縮方法では、ＣＰＵの命令の性質によって、その後に続くマシンコードが所定の配置となる、いわゆるＣＰＵのくせを利用して圧縮している。このため、プログラムの内容に拘わらずそのプログラムを高圧縮率なものとすることができる。

【００８０】また、請求項３記載のプログラム復号方法では、ＣＰＵの命令の性質によって、その後に続く圧縮されたマシンコードが所定の配置となる、いわゆるＣＰＵのくせを利用して復号している。このため、圧縮されたプログラムの内容に拘わらずそのＣＰＵにとって最適かつ高効率にて復号することができる。

【００８１】さらに、請求項４から６記載のプログラム格納装置では、ＣＰＵの命令の性質によって、その後に続くマシンコードが所定の配置となる、いわゆるＣＰＵのくせを利用して圧縮したり復号したりしている。このため、プログラムをそのＣＰＵにとって最適かつ高効率で圧縮したり復号したりできる。加えて、記憶装置の容量が減少し、低価格化が可能となると共に、同じ容量の記憶装置を使用すると、大容量のプログラムを保存することが可能となる。

【図面の簡単な説明】

【図１】本発明の第１の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図２】図１のプログラム格納装置に使用されるＣＰＵの命令タイプを示す図で、（Ａ）から（Ｐ）の計１６種類のタイプに区分されることを示す図である。

【図３】図１のプログラム格納装置に使用されるＣＰＵの命令タイプと、その命令タイプのときに使用する圧縮器との関係を示す図である。

【図４】図１のプログラム格納装置で使用されるプログ

17

ラム圧縮方法を説明するためのフローチャートである。

【図５】図１のプログラム格納装置および図４のフローチャートによって対象プログラムが圧縮された結果を示す図で、参考に従来の方法による圧縮率も併せて示した図である。

【図６】図１のプログラム格納装置および図４のフローチャートによって対象プログラムを圧縮する際のコンテキストの利用状況を説明するための図である。

【図７】本発明の第２の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図８】本発明の第３の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図９】従来の算術符号型のエントロピー符号器およびエントロピー復号器の説明図である。

【図１０】従来の符号化システムおよび復号化システムのブロック図である。

【図１１】従来の状態信号の生成を説明するための図で、符号化対象画素データに対する参照画素データの説明図である。

18

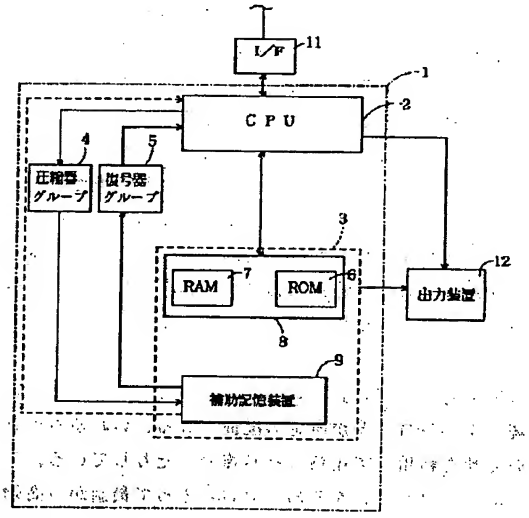
【図１２】状態縮退器を有する従来の符号化システムおよび復号化システムのブロック図である。

【図１３】従来の最新出現表の使用例を説明するための図で、（Ａ）は入力されてくる画素と符号化対象画素の位置関係を示し、（Ｂ）は、各画素のカラーシンボルを示し、（Ｃ）は最新出現表を示す図である。

【符号の説明】

1　プログラム格納装置
2　ＣＰＵ（中央処理装置）
3　記憶装置
4　圧縮器グループ
5　復号器グループ
6　ＲＯＭ（リード・オンリ・メモリ）
7　ＲＡＭ（ランダム・アクセス・メモリ）
8　主記憶装置
9　補助記憶装置

【図１】



【図３】

○：使用を示す

命令タイプ	使用する圧縮器			
	命令フィールド用	レジスタフィールド用	条件フィールド用	定数フィールド用
Aグループ	○	○		
Bグループ	○	○		○
Cグループ	○		○	○
Dグループ	○			○
Eグループ	○	○		○
Fグループ	⋮	⋮	⋮	⋮
Iグループ	○			○
⋮	⋮	⋮	⋮	⋮

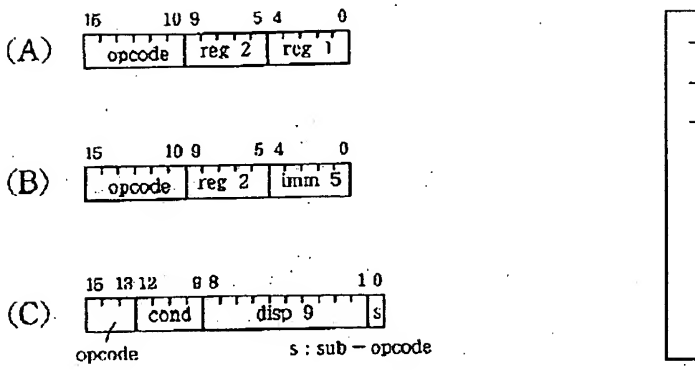
【図５】

方式		対象プログラム（2,000バイト） 圧縮率
参考	LZ系基本アルゴリズム	69.0％
本発明	4＋5＋4＋1ビットQMコーダ	75.0％
	コンテキスト採用 4＋5＋4＋1ビットQMコーダ	69.6％

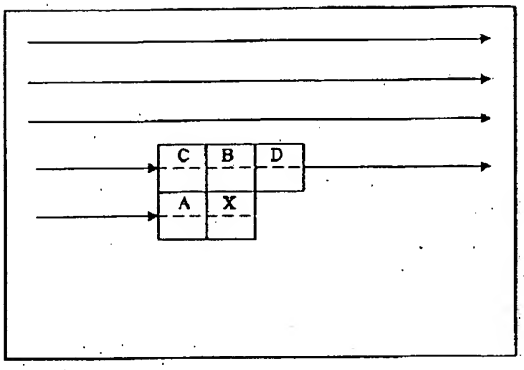
【図６】

	コンテキスト（過去の単純データ）	
命令フィールド	4ビット×0個	※コンテキスト利用しない
レジスタフィールド	5ビット×2個	直前2個分の確率統計を取る
条件フィールド	4ビット×0個	※コンテキスト利用しない
定数フィールド	1ビット×9個	単純に過去9個分を

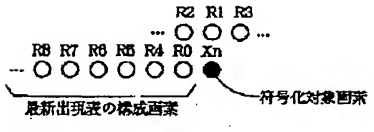
【図２】



【図１１】



【図１３】

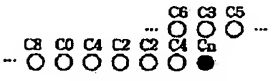


画素優先順位

学習なし：R0,R1,R2…R8…（全画素固定）

学習機能：R0,R1,R2,R3, R4,R5…R8…

学習により可変　固定



Cn：カラーシンボル

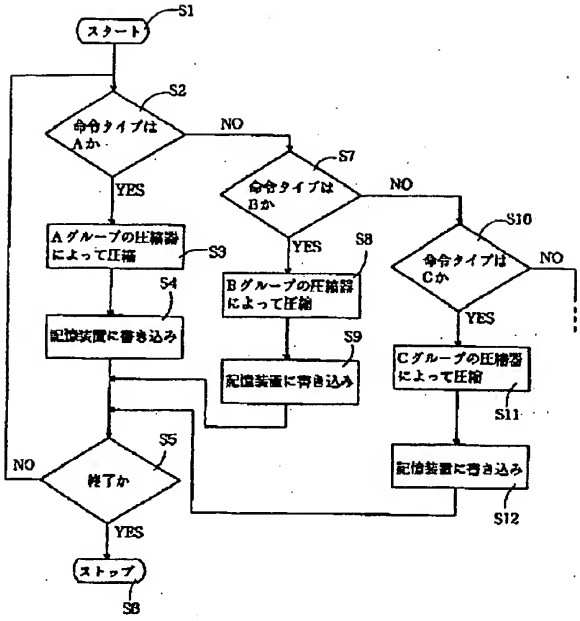
(n：0～15)

(C)

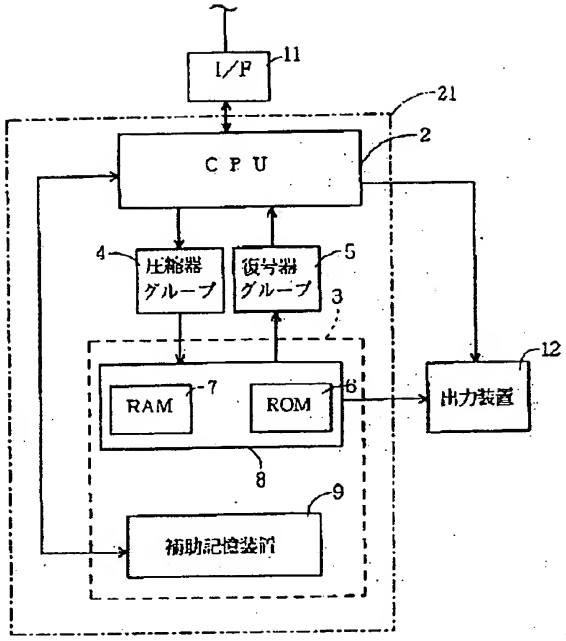
最新出現表

色順位	カラーシンボル
1位	C4
2位	C8
3位	C6
4位	C5
5位	C2
6位	C0
7位	C8
⋮	⋮
16位	Cn

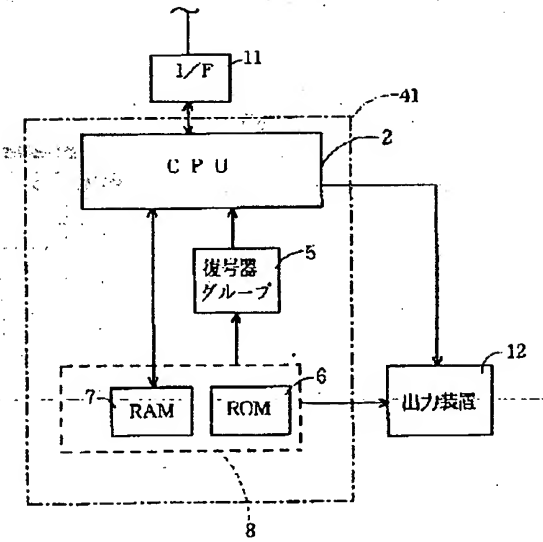
【図４】


スタート
S1
命令タイプはAか
S2
NO
YES
命令タイプはBか
S7
NO
YES
命令タイプはCか
S10
NO
YES
Aグループの圧縮器によって圧縮
S3
Bグループの圧縮器によって圧縮
S8
Cグループの圧縮器によって圧縮
S11
記憶装置に書き込み
S4
S9
S12
終了か
S5
NO
YES
ストップ
S6

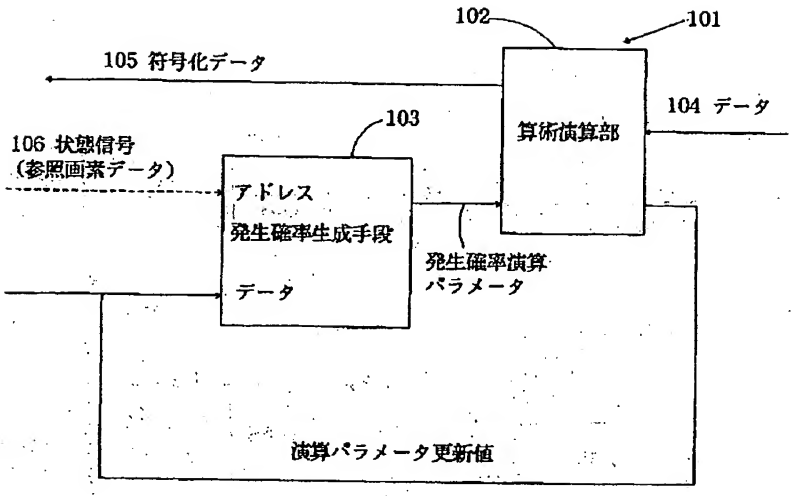
【図７】


I/F
11
21
CPU
2
圧縮器グループ
4
復号器グループ
5
3
RAM
7
ROM
6
出力装置
12
8
補助記憶装置
9

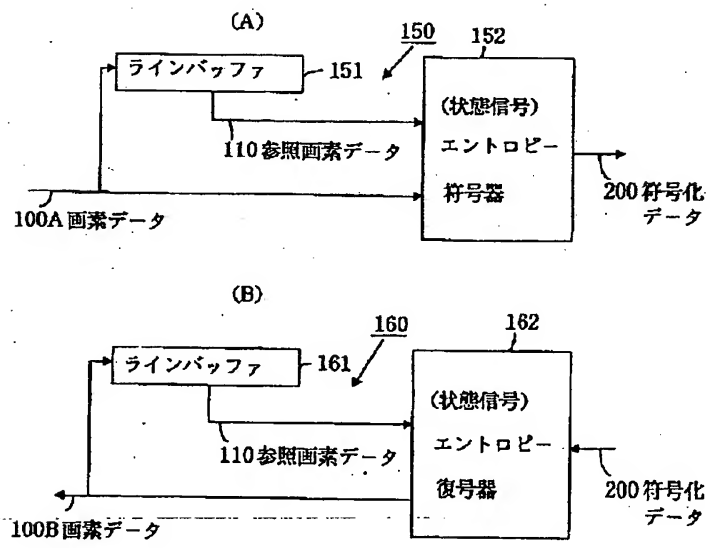
【図８】


I/F
11
41
CPU
2
復号器グループ
5
RAM
7
ROM
6
出力装置
12
8

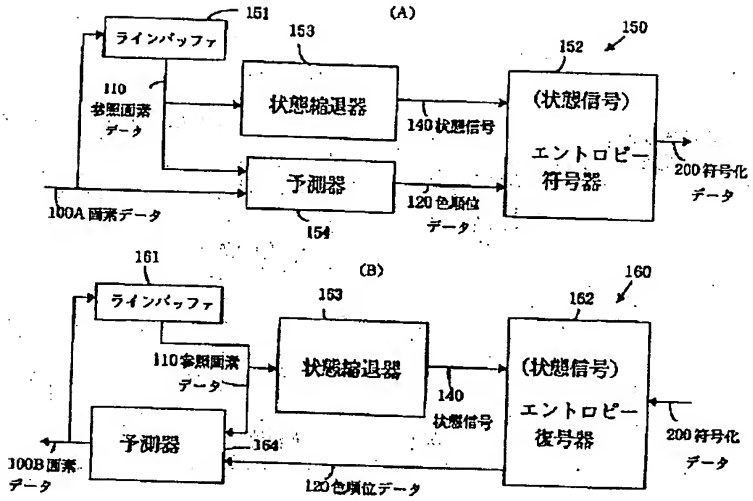
【図９】


102
101
105 符号化データ
104 データ
算術演算部
103
106 状態信号
(参照画素データ)
アドレス
発生確率生成手段
データ
発生確率演算
パラメータ
演算パラメータ更新値

【図１０】


(A)
150
152
ラインバッファ
151
(状態信号)
エントロピー
符号器
110参照画素データ
100A画素データ
200符号化
データ
(B)
160
162
ラインバッファ
161
(状態信号)
エントロピー
復号器
110参照画素データ
100B画素データ
200符号化
データ

【図１２】


(A)
151
ラインバッファ
153
状態縮退器
150
152
(状態信号)
エントロピー
符号器
200符号化
データ
110
参照画素
データ
140状態信号
予測器
120色順位
データ
154
100A画素データ
(B)
161
ラインバッファ
163
160
162
(状態信号)
エントロピー
復号器
200符号化
データ
110参照画素
データ
状態縮退器
140
状態信号
予測器
164
100B画素
データ
120色順位データ